Efficient Cross-Datacenter Congestion Control with Fast Control Loops

Baosen Zhao[†] [◊], Jianan Sun[†], Xu Zhou[†◊*], Wanghong Yang[†], Wenji Du[†] [◊] Fukang Chen[†] [◊], Yongmao Ren[†] [◊], Stefan Schmid [‡]

[†] Computer Network Information Center, Chinese Academy of Sciences

[◊] University of Chinese Academy of Sciences

[‡] TU Berlin

Abstract

Many applications, such as AI training and distributed storage, rely on cross-datacenter (DC) networks to provide services. For compatibility with existing RDMA hardware and to improve quality of service, network providers connect to datacenters over dedicated lines. However, the current RDMA congestion control has some problems in cross-DC environment. First, due to the large bandwidth delay product (BDP) of cross-DC flows, the switch frequently triggers PFC, which impairs the transmission of all flows. Meanwhile, due to the lag of congestion signals, congestion control algorithms may cause unfair bandwidth allocation between intra-DC flows and cross-DC flows. In addition, cross-DC traffic will experience severe queuing at the data center interconnect (DCI) switch, increasing queuing delay. To address these challenges, this paper proposes MLCC, a cross-datacenter congestion control algorithm based on the fast control loop. MLCC uses micro congestion control loops to achieve fine-grained network state awareness and accurate rate adaptation, and reduce queue length in the transmission path. Experimental results show that MLCC can quickly converge all flows to fairness, achieve high link utilization, and ensure low queue length on the switch. Large-scale simulations show that MLCC can reduce the average FCT of intra-datacenter and cross-datacenter traffic by up to 46% and 27%, respectively.

Keywords

Cross-datacenter Communication, Congestion Control, Transmission Optimization, RDMA

ACM Reference Format:

Baosen Zhao[†] [◊], Jianan Sun[†], Xu Zhou[†◊*], Wanghong Yang[†], Wenji Du[†] [◊] Fukang Chen[†] [◊],, Yongmao Ren[†] [◊], Stefan Schmid [‡]. 2018. Efficient Cross-Datacenter Congestion Control with Fast Control Loops. In *Proceedings of The International Conference on Parallel Processing 2025 (ICPP'2025).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*Corresponding author.

1 Introduction

The rapid expansion of cloud computing, artificial intelligence (AI), and other emerging technologies has driven enterprises to seek more advanced datacenter (DC) infrastructures to accommodate their growing demands for computing power and storage capacity [1, 7, 18]. However, a single DC is no longer adequate to meet the increasing requirements for data processing and storage services. In response, many enterprises are establishing multiple geographically distributed datacenters, interconnected through dedicated communication links to provide unified, seamless services to end-users [6]. With the growth of cross-DC networks, the need for high-speed data transfers over DC interconnection (DCI) networks is becoming increasingly important[13].

Remote Direct Memory Access (RDMA) network interface card (NIC) has become a popular choice for high-speed networking hardware in DC, due to their advantages of high throughput, low latency, and minimal CPU overhead [8]. While existing RDMA congestion control algorithms are effective for intra-DC communication, their direct application to cross-DC scenarios can significantly degrade inter-DC transmission performance. This is because most RDMA congestion control mechanisms are designed for the low-latency environment typical of intra-DC networks [32]. In contrast, inter-DC communication often involves long-distance dedicated network links, which introduce higher end-to-end latency [33]. This additional latency can cause RDMA congestion control algorithms to misinterpret network conditions [24], resulting in several detrimental effects on both intra-DC and inter-DC networks: (1): Unfair bandwidth allocation between intra-DC and inter-DC traffic; (2): Increased susceptibility to triggering Priority Flow Control (PFC), leading to link pauses; (3): Higher queuing delays at DCI-switches. Consequently, the development of efficient congestion control algorithms for cross-DC networks has become one of the most significant challenges.

To address this challenge, cross-DC networks need to choose the right transmission protocols that allow both intra-DC and inter-DC traffic to coexist efficiently. However, neither DCQCN [32] nor the latest BICC [28] congestion control algorithms fully meet these needs. First, DCQCN relies on end-to-end feedback, which causes long RTT inter-DC flows to take up too much bandwidth, pushing aside short RTT intra-DC flows. Second, the BICC algorithm groups flows with the same destination into a single queue at the receiver-side DCI, which harms fairness between different flows. Lastly, BICC doesn't consider the queue length at DCI-switches, leading to longer queuing delays at DCI-switches.

In recent years, DCI-switches have introduced new features to meet user requirements, such as programmable packet processing

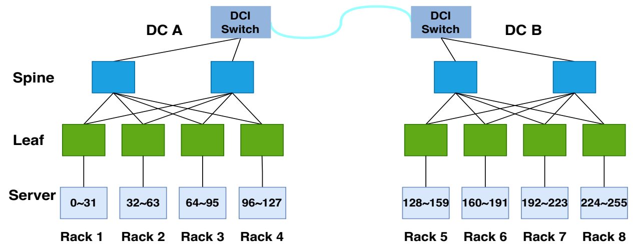


Figure 1: The overview of cross-datacenter network system.

and virtual queue management [26]. For example, Swing [5] uses DCI-switches to relay PFC signals, and BiCC [28] leverages DCI-switches to regulate the transmission rate of cross-DC traffic. In addition, in-band network telemetry (INT) can accurately monitor network performance in real time, allowing the server to adjust the transmission rate based on accurate network status information [2, 4, 16]. Therefore, the introduction of these features makes DCI-switches play a more important role in cross-DC network.

Based on the above analysis, this paper introduces MLCC. MLCC leverages the unique network characteristics of DCI-switches, RDMA, and INT to optimize the throughput of inter-DC traffic while maintaining low latency for intra-DC traffic. The MLCC algorithm avoids excessive global coordination through multiple feedback loops. It uses near-source feedback and receiver-driven loops to independently adjust intra-DC rates in the short term, enabling rapid congestion mitigation. Additionally, MLCC employs end-to-end feedback to coordinate inter-DC rates, ensuring long-term rate stability and effective DCI-switch queue management.The main contributions of our work are summarized as follows:

(1) We introduce a novel cross-datacenter transmission mechanism leveraging DCI-switch coordination. The DCI-switch segments the long-distance end-to-end control into multiple precise RDMA congestion control loops by utilizing INT information and queue management services. This mechanism enables effective flow rate adaptation between cross-datacenter and intra-datacenter traffic.
(2) We develop a receiver-driven congestion control algorithm based on INT. This algorithm dynamically calculates and regulates the dequeue rate at the DCI-switch using INT information from the intra-datacenter switch, ensuring minimal queuing in the receiver-side datacenter.
(3) We present the DCI-switch queue management (DQM) algorithm, which leverages INT information from the DCI-switch to predict queuing delays. By providing feedback to the sender on the appropriate sending rate, this algorithm efficiently reduces queues at the DCI-switch.

2 Motivation

In this section, we first introduce the background of cross-datacenter transmission systems. Then, we analyze the impact of cross-datacenter flow on bandwidth and latency. Finally, we discuss the motivation for developing the MLCC framework.

2.1 Background of Cross-Datacenter

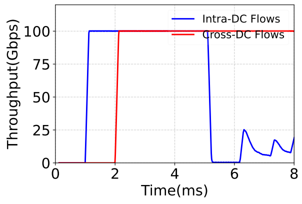
In datacenter network, remote direct memory access (RDMA) technology bypasses the traditional TCP/IP protocol stack to directly implement zero-copy data interaction between server memories, reducing end-to-end communication latency to microseconds and significantly improving throughput efficiency. As the core protocol of RDMA deployment in Ethernet, RDMA over Converged Ethernet (RoCE) builds a lossless transmission environment based on the priority flow control (PFC) mechanism defined in the IEEE 802.1Qbb standard [25]. PFC implements precise flow control through dual thresholds (Xoff/Xon). Through dynamic coordination with 8 priority queues, the datacenter switch can buffer burst traffic impacts and maintain sub-microsecond transmission latency.

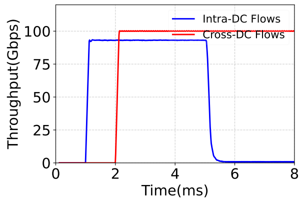
As cloud service providers extend RDMA to regional deployments, cross-DC network architectures need to coordinate the co-design of intra-DC and inter-DC networks. As shown in Fig. 1, cross-DC traffic follows the path: from the sending server through the local egress DCI-switch, long-distance fiber links to the remote datacenter ingress DCI-switch, and finally to the receiving server [12–14]. To achieve seamless communication, cloud service providers use dedicated optical fibers to build wide-area backbone networks and deploy DCI-switches that support 100-400 Gbps and are equipped with hundreds of MB of cache [30]. Its large cache design is specifically designed to cope with the challenges of burst traffic and propagation delay in long-distance transmission. In contrast, high-speed switches of intra-DC are only equipped with an MB-level buffer to ensure μs-level low-latency communication within the datacenter. This high-performance infrastructure provides a reliable hardware foundation to meet the growing demand for cross-DC transmission.

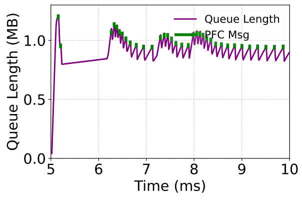
With the rise of cross-DC architectures, network traffic can be classified into intra-DC and inter-DC traffic. Intra-DC traffic, which involves communication between servers within the same DC, requires sub-microsecond latency for tasks like AI training and distributed storage. Inter-DC traffic, exchanged between geographically separated DCs, demands higher bandwidth for large-scale data transfer. Studies show that the ratio of intra-DC to inter-DC traffic is about 6:1 [9]. This poses a challenge to network resource allocation: ensuring low-latency transmission for intra-DC traffic and sufficient bandwidth for inter-DC traffic to balance latency-sensitive and bandwidth-sensitive requirements.

2.2 Limitations of Existing Datacenter Congestion Control

In current datacenter networks, RDMA congestion control mechanisms typically rely on explicit congestion notification (ECN) or round-trip time (RTT) as congestion signals [20, 29]. For instance, DCQCN is a representative ECN-based protocol where the sender dynamically adjusts its transmission rate based on ECN marks fed back by the receiver, effectively reducing queue length and latency within the datacenter network [32]. Recently, some researchers have proposed receiver-driven congestion control approaches, such as NDP [10] and FlexPass [17], demonstrating extremely low queuing delays. However, these approaches are not directly applicable to cross-datacenter communication. In the following, we conduct experimental investigations to illustrate the limitations of current


(a) Throughput of DCQCN


(b) Throughput of PowerTCP


(c) Queue length and PFC triggering time with DCQCN

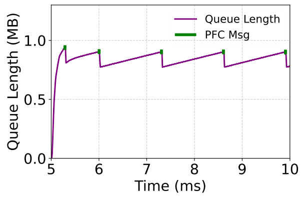

(d) Queue length and PFC triggering time with PowerTCP

Figure 2: Throughput and queue length of DCQCN and PowerTCP algorithms, when the congestion point is in receiver-side DC.

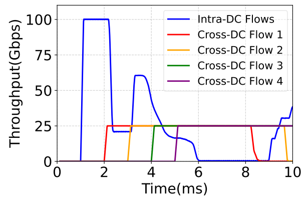
congestion control protocols in cross-datacenter transmissions. The experimental topology is illustrated in Fig. 1. For clarity, we have numbered the servers and racks. The propagation delay of long-distance links is 3 ms.

PFC is frequently triggered, causing damage to all datacenter flows passing through the bottleneck queue. Due to the longer RTT of cross-datacenter flows, their Bandwidth-Delay Product (BDP) significantly exceeds the buffer capacity of typical intra-DC switches [24], making it more likely to trigger PFC and limit the transmission of intra-DC flows.

In Experiment 1, at 1 ms, four servers in Rack 5 begin sending data to four servers in Rack 6. At 2 ms, four additional servers in Rack 1 also start sending data to the servers in Rack 6. Fig. 2 illustrates the throughput of different flows, along with the queue length of the bottleneck link switch and the PFC trigger times. At 5 ms, we can see that when the cross-datacenter flow reaches the receiver-side datacenter, the queue length of the switch in the datacenter increases sharply and triggers PFC. This occurs because the large volume of cross-datacenter traffic overwhelms the shallow buffers of the datacenter switches, quickly reaching the PFC trigger threshold.

Unfairness between Intra-Datacenter and Cross-Datacenter Traffic. The existing RDMA congestion control algorithms rely on end-to-end congestion signal feedback. For intra-datacenter traffic, RTT is typically around 20 to 100 μs [20], while for cross-datacenter traffic, RTT ranges from 2 to 6 milliseconds, which is several hundred times higher. This significant difference in RTT can result in unfair bandwidth allocation between intra-datacenter and cross-datacenter flows.

In Experiment 2, at 1 ms, four servers in Rack 1 initiate communication with four servers in Rack 2 within the same datacenter. At 2 ms, another four servers in Rack 1 sequentially initiate communication with servers in Rack 5 located in another datacenter. As shown in Fig. 3, the results indicate that as the number of cross-datacenter


(a) DCQCN

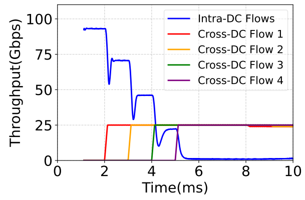
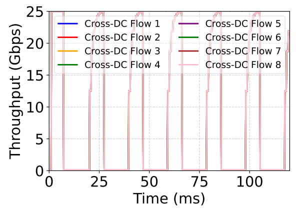

(b) PowerTCP

Figure 3: Throughput of DCQCN and PowerTCP algorithms, when the congestion point is in sender-side DC.


(a) Throughput

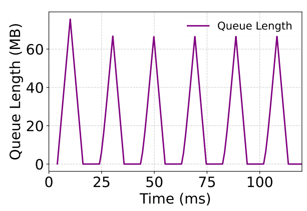

(b) Queue length

Figure 4: Throughput and queue length of receiver-side DCI-Switch.

flows increases, the available bandwidth for intra-datacenter flows is progressively occupied, leading to significant performance degradation. This phenomenon occurs because intra-datacenter traffic with shorter RTT can detect network congestion more quickly, rapidly reduce its transmission rate, and ultimately bear the full burden of clearing the queue, resulting in unfair bandwidth allocation.

Cross-datacenter flows experience significant queuing at DCI-switches. Due to the large buffer capacity of DCI-switches, congestion signals are not easily triggered even when a substantial amount of cross-datacenter traffic is buffered. As a result, cross-datacenter flows are prone to long queuing at DCI-switches.

In Experiment 3, at 1 ms, four servers in Rack 1 start transmitting data packets to a server in Rack 6, while four servers in Rack 4 simultaneously send data to the same server in Rack 6. Fig. 4 illustrates the throughput of each data flow and the queue length at the receiver-side DCI-switch. It can be observed that the throughput of each data flow fluctuates continuously, while the queue length at the receiver-side DCI-switch first increases to a high level, gradually decreases, and then repeats this process. This behavior occurs because when the combined throughput of these data flows exceeds the egress rate of the receiver-side DCI-switch, the excess data is buffered until the queue length reaches the ECN threshold, triggering a congestion signal that is fed back to the senders. Consequently, cross-datacenter flows experience significant queuing at DCI-switches.

2.3 Can We Further Improve Cross-DC Transmission?

The experiments above demonstrate that the existing cross-datacenter transmission system faces the following three key challenges:

(1) Due to the significant difference in RTT between cross-datacenter and intra-datacenter traffic, cross-datacenter flows can severely monopolize the bandwidth of intra-datacenter flows.
(2) The rapid influx of large volumes of cross-datacenter traffic into the datacenter network can overwhelm the shallow-buffered datacenter switches, frequently triggering PFC.
(3) Traditional congestion signals, such as ECN, fail to accurately reflect the switch's status, making it difficult for the host to adjust the sending rate precisely and eliminate queuing delays at the switch.

Based on the above analysis, we argue that achieving efficient cross-datacenter communication requires improvements in the following areas:

(1) Establishing the shorter congestion control loop to ensure that all hosts can timely acquire network status information.
(2) Managing cross-datacenter traffic at the DCI-switch to control the enqueue rate.
(3) Providing a new congestion signal that accurately reflects switch status to guide precise rate adjustments at the host.

To address these challenges, we propose Micro Loop Congestion Control (MLCC), a congestion control protocol designed for cross-datacenter communication, with the primary objective of achieving a fast control loop.

3 Micro Loop Congestion Control

In this section, we first present the framework of the Micro Fast Loop Congestion Control (MLCC) algorithm. We then detail the design components of MLCC, followed by the complete algorithm and overall system architecture.

3.1 Overview

By deeply analyzing the characteristics of cross-datacenter traffic, we propose a mechanism based on minimized control loops, Minimized Loop Congestion Control (MLCC). This mechanism independently adjusts the traffic rate of intra-DC in the short term by adopting near-source feedback and receiver-driven control loops, thereby achieving rapid congestion mitigation. On this basis, MLCC also introduces an end-to-end feedback mechanism to coordinate rates between datacenters, ensure long-term rate stability, and effectively manage the queues of datacenter interconnect (DCI) switches.

As shown in Fig. 5, MLCC consists of two parts: the short-term control loop for intra-DC and the long-term control loop across dataceners. Through these two feedback mechanisms, MLCC can optimize traffic control both locally and globally, thus achieving efficient traffic management and scheduling.

Short-term control loops of intra-DC: The short-term control loop of intra-DC is further subdivided into the near-source feedback loop and the receiver-driven control loop.

(1) **DCI-switch near-source feedback loop:** The switches between the sender and the sender-side DCI-switch insert INT information into the data packet. The sender-side DCI-switch processes the INT information from the data packet and sends it back to the sender in the feedback packet. In this way, MLCC quickly alleviates congestion caused by mixed traffic in the sender-side datacenter.
(2) **Receiver-driven control loop:** The congestion control loop driven by the receiver uses a credit-driven algorithm to calculate the dequeue rate of the DCI-switch. MLCC can effectively alleviate the congestion of the receiver-side datacenter through this loop.

End-to-end control loop: The end-to-end control loop makes a holistic decision by integrating feedback from the two control loops and the queue management algorithm of the DCI-switch. It then adjusts the sending rate accordingly, effectively reducing the queue length at the DCI-switch.

3.2 Short-term Control Loops

3.2.1 DCI-switch Near-source Feedback Loop.

The data packet will pass through multiple switches from the sender server to the sender-side DCI-switch. These switches will insert INT information into the data packet. If end-to-end feedback is used for cross-datacenter transmission, the INT information in the data packet needs at least one cross-datacenter RTT RTT_C to reach the sender for congestion control. This untimely feedback will cause unfairness between intra-datacenter traffic and cross-datacenter traffic. Since the DCI-switch is close to the sender, we propose the DCI-switch near-source feedback loop. When the DCI-switch receives a cross-datacenter data packet, it reads the INT information and clears the INT information in the data packet. The DCI-switch encapsulates the INT information into the Switch-INT packet through the CNP message and then sends it to the sender [31]. The sender calculates the fair rate of the flow in the sender-side datacenter based on the INT information in the Switch-INT packet. Through the DCI-switch near-source feedback loop, the sender can effectively alleviate the network congestion in the sender-side datacenter.

3.2.2 Receiver-driven Control Loop.

To limit the network congestion caused by mixed traffic in the receiver-side datacenter, MLCC further shortens the control loop in the receiver-side datacenter through the receiver-driven control loop. The receiver-driven control loop consists of the DCI-switch scheduling mechanism and the credit-driven algorithm.

DCI-switch scheduling mechanism: When a cross-datacenter flow arrives at the receiver-side datacenter, the receiver-side DCI-switch erases and reinserts the INT information. MLCC uses dynamically allocated Per-Flow Queuing (PFQ) [19] to store traffic of the same flow on the receiver-side DCI-switch. Specifically, the basic principle of PFQ is that each flow has a corresponding virtual queue at the switch. In this way, network devices can manage their packets separately for each traffic flow, thereby improving fairness and isolation. The receiver-side DCI-switch first checks if the flow has been assigned a PFQ when a packet arrives. If so, the DCI-switch pushes the packet into the relevant PFQ. Otherwise, an empty PFQ is assigned. For a new PFQ, the receiver-side DCI-switch sends the flow into the receiver-side datacenter using the initial rate.

Credit-driven algorithm: The INT information in the data packet arriving at the receiver server contains the INT information of the receiver-side DCI-switch and the switch in the receiver-side

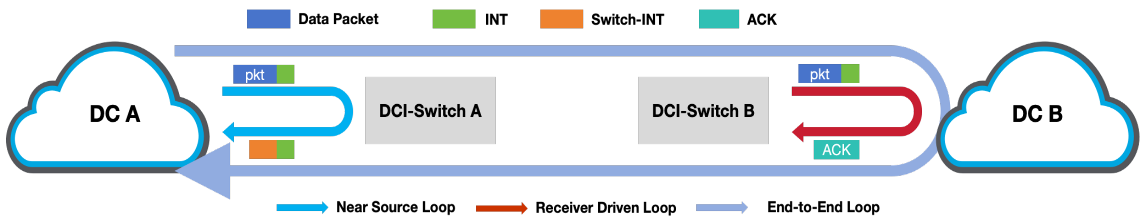


Figure 5: The framework of MLCC.

Algorithm 1: Credit-driven Algorithm

```
Input  : New flow $F$ arriving at receiver-side datacenter
Output : Dequeue Rate $R_{credit}$
1  Initialization:
2  Receiver initializes the credit value $C_R$ for the new flow $F$ ;
3  Receiver-side DCI-switch reads $C_R$ from the ACK packet ;
4  Update the credit value $C_D$ record for PFQ in the
     DCI-switch ;
5  Receiver-side DCI-switch inserts $C_D$ into subsequent data
     packets;
6  Rate Control Stage:
7  while Data packet from flow $F$ do
8      Read credit value $C_D$ from the data packet;
9      if $C_R == C_D$ then
10         $C_R$++;
11         Update congestion control parameters;
12         Calculate dequeue rate $R_{credit}$ for the PFQ;
13         Insert new $C_R$ and $R_{credit}$ into the ACK packet;
```

datacenter. To alleviate the congestion of the receiver-side datacenter and shorten the control loop, we implemented the credit-driven algorithm at the receiver server. The receiver server can control the dequeue rate of the receiver-side DCI-switch through the credit-driven algorithm. Through the credit-driven algorithm, MLCC achieves fairness of mixed flows in the receiver-side datacenter. Algorithm 1 depicts the credit-driven algorithm.

In the initial stage, the receiver initializes the credit value C_R when it receives a new flow. The receiver-side DCI-switch reads C_R in the ACK packet to update the credit value C_D for the PFQ. And the receiver-side DCI-switch will insert C_D in the subsequent data packets.

In the rate control stage, the receiver compares the credit value C_R with the C_D value in the data packet when the receiver receives the data packet. If C_R and C_D are the same, the flow has sent one datacenter RTT. The receiver will update the C_R, the relevant parameters of the congestion control algorithm, and calculate the dequeue rate R_{credit} for the PFQ of this flow. The receiver inserts the new C_R and dequeue rate R_{credit} in the ACK. Therefore, the receiver-side DCI-switch can read the C_R and dequeue rate R_{credit} in the ACK packet to control the dequeue rate of PFQ.

Table 1: VARIABLES AND PARAMETERS

Symbol	Definition
RTT_D	RTT of intra-DC
RTT_C	RTT of cross-DC
R_{credit}	Dequeue rate of receiver-side DCI-switch
R_{pre_eq}	Predicted average enqueue rate
R_{DQM_i}	DQM rate of the i-th RTT_D

3.3 Long-term Control Loop

3.3.1 *DCI-switch Queue Management Module.*

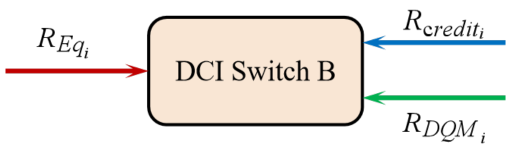
As described in Section 2.2 , cross-DC traffic is prone to long queues on the receiver-side DCI-switch, increasing the end-to-end transmission delay. As analyzed above, the queue length of the DCI-switch will affect the end-to-end delay. Therefore, we propose the DCI-switch Queue Management (DQM) algorithm to reduce the queue length of the receiver-side DCI-switch. By leveraging the INT information of the receiver-side DCI-switch, the receiver can accurately obtain the queue length and then reduce the queue length with the DQM algorithm.

The core idea of the DQM algorithm is to predict the BDP under the RTT_D scale, so that the BDP of the flow does not exceed the sum of the BDP of the link and the DCI-switch queue length threshold. Consider the cross-datacenter link as a horizontal water pipe, where the length of the pipe is

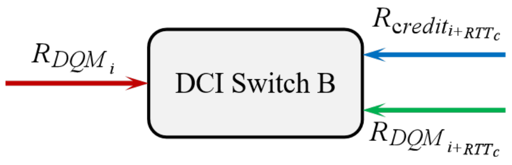
$$n = \frac{RTT_C}{RTT_D} \tag{1}$$

and the cross-sectional area represents the rate at each RTT_D. DQM must satisfy two key requirements: (1): Given the limited buffering capacity of the intra-DC switch, the goal for cross-DC flows is to maintain the average BDP below the buffering threshold. (2): Minimize the queue length at the receiver-side DCI-switch.

As shown in Fig. 6, the receiver-side DCI-switch can obtain the enqueue rate R_{Eq_i}, dequeue rate R_{credit_i}, and end-to-end suggestion rate R_{DQM_i} of the i-th RTT_D where the RTT_D represents the RTT of intra-datacenter link. We observe that when the cross-datacenter traffic reaches a stable state, the enqueue rate at time $i+RTT_C$ is the end-to-end suggestion rate R_{DQM_i} where RTT_C represents the RTT of cross-datacenter end-to-end link. This means that the receiver-side DCI-switch can know the enqueue rate of the next RTT_C in advance. Through this feature, the receiver-side DCI-switch can predict the queue length of the next RTT_C and control the queue length.



(a) The state of the DCI-switch at time t1



(b) The state of the DCI-switch at time t1+RTT_C

Figure 6: The state of the receiver-side DCI-switch.

The difference between the enqueue rate and the dequeue rate causes queue accumulation. Therefore, to predict the queue length, we first need to predict the enqueue and dequeue rates on the RTT_C time scale. We can obtain the enqueue rate on the RTT_C time scale by averaging the enqueue rates on multiple RTT_D time scales. As analyzed before, R_{DQM_i} can be used as the predicted value of the enqueue rate at the $i + RTT_C$ time. So the predicted enqueue rate on the RTT_C time scale is calculated as follows:

$$R_{pre\_eq} = \frac{\sum_{i-n-1}^{i-1} R_{DQM_i}}{n} \tag{2}$$

where R_{pre_eq} is the predicted average enqueue rate of the cross-datacenter flow on the RTT_C time scale. $\sum_{i-n}^{i} R_{DQM_i}$ represents the sum of n R_{DQM}. n is the ratio of RTT_C to RTT_D.

Through the above credit-driven algorithm, we can obtain the dequeue rate. So the predicted queue length of the next RTT_C is calculated as follows:

$$Q_{pre} = (R_{pre\_eq} - R_{credit}) \cdot RTT_C + Q_c \tag{3}$$

where Q_{pre} is the predicted queue length at the next RTT_C. R_{credit} is the dequeue rate at the current time. Q_c is the current queue length of PFQ for the cross-datacenter flow. By predicting the queue length and considering the current dequeue rate, we can estimate the queuing delay at the current dequeue rate. Therefore, the predicted queuing delay is calculated as follows:

$$D_{pre} = m \cdot \frac{Q_{pre}}{\sum_{i-m}^{i} R_{credit_i}} \tag{4}$$

where D_{pre} represent the predicted queuing time. To prevent abrupt changes in R_{credit_i}, we smooth it by averaging the most recent m values of R_{credit_i}.

However, it is not necessary to fully empty the DCI-switch queue. Instead, the goal is to maintain the queue length within a certain range to reduce the queuing time of packets at the DCI-switch. Increasing the dequeue rate at the DCI-switch would disrupt the fairness among mixed flows at the receiver-side datacenter. Therefore, we manage the queue length by controlling the end-to-end rate. The target end-to-end rate is calculated as follows:

$$R_{DQM_i} = R_{credit_i} \cdot \left(1 - \frac{D_{pre} - D_t}{\theta}\right) \tag{5}$$

Algorithm 2: DQM Algorithm

```
Input: $R_{credit}$, $RTT_C$, $RTT_D$
Output: Suggestion End-to-End Rate
1  $n \leftarrow \frac{RTT_C}{RTT_D}$;
2  $R_{pre\_eq} \leftarrow Eq$ (2);
3  $Q_{pre} \leftarrow Eq$ (3);
4  $D_{pre} \leftarrow Eq$ (4);
5  if $\overline{T_{pre}} > T_q$ then
6      $R_{DQM} \leftarrow Eq$ (5);
7  while $token < 1$ do
8      $token \leftarrow token + \min\left(\alpha \cdot \frac{R_{DQM}}{R_{credit}}, 1\right)$;
9  if $token > 1$ then
10     $dw \leftarrow dw + 1$ ;
11 else
12     $dw \leftarrow dw - 1$;
13 $\overline{R_{DQM}} \leftarrow Eq$ (9);
14 return $\overline{R_{DQM}}$;
```

where R_{DQM_i} represents the end-to-end rate of DQM algorithm of current RTT_D. D_t represents the threshold of queuing delay. θ represents the time required to transform the queuing delay from D_{pre} to D_t. When the queue delay exceeds D_t, maintaining the rate R_{DQM} for the duration of θ will reduce the queuing delay to D_t. In order to maintain the stability of the end-to-end rate, it is recommended that the threshold for θ be set greater than RTT_C. However, network jitter can cause drastic changes in R_{DQM}, so we design a smoothing mechanism.

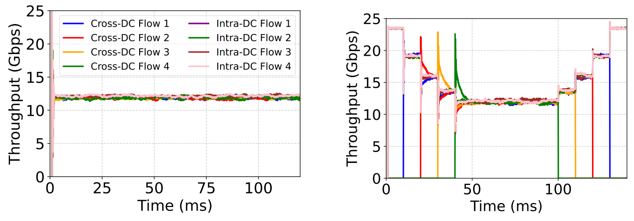
DQM employs a token bucket mechanism to effectively limit burstiness. As described in Eq.6, the token bucket increments the token number with each outgoing packet. When the token number exceeds 1, it signifies an increase in transmission rate based on R_{credit}. Conversely, when the token number is below 1, it indicates a decrease in rate relative to R_{credit}. This simple mechanism effectively ensures smooth and gradual rate adjustments.

$$token = \begin{cases} token, & \text{if } token < 1 \\ token - 1, & \text{if } token \geq 1 \end{cases} \tag{6}$$

where:

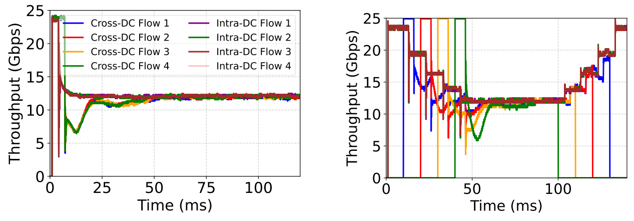
$$token = token + \min\left(\alpha \cdot \frac{R_{DQM}}{R_{credit}}, 1\right) \tag{7}$$

As described in Eq.8, dw is the dynamic window that updates in response to the current token value. When the token value exceeds 1, dw is incremented, allowing the system to handle higher rates. Conversely, when the token is less than 1, dw is decremented to reflect the need to lower the transmission rate. $\overline{R_{DQM}}$ represents the smoothed rate, which is calculated based on the baseline rate R_{credit} and the influence of dw while also taking into consideration the Maximum Transmission Unit (MTU) and the congestion round-trip time (RTT_C). This measure enables fine-grained rate adjustment by dynamically integrating congestion feedback and overall network conditions, ensuring consistent end-to-end transmission efficiency



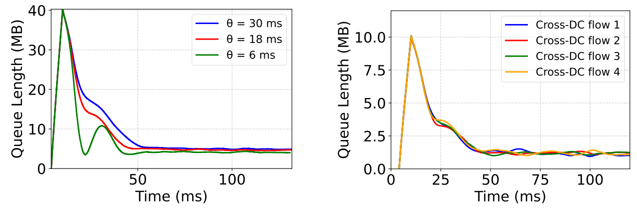
(a) Simultaneously **(b) Sequentially**

Figure 7: Convergence of MLCC When the the bottleneck link in sender-side datecenter.



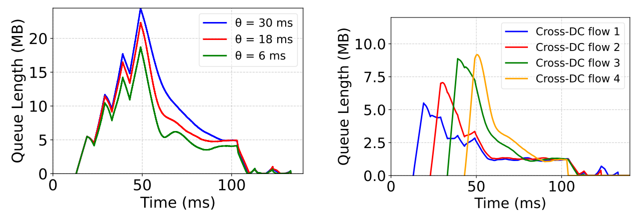
(a) Simultaneously **(b) Sequentially**

Figure 8: Convergence of MLCC When the the bottleneck link in receiver-side datecenter.



(a) Buffer length **(b) Queue length for each flow when θ = 18 ms**

Figure 9: The queue length of receiver-side DCI-switch when simultaneous burst.



(a) Buffer length **(b) Queue length for each flow when θ = 18 ms**

Figure 10: The queue length of receiver-side DCI-switch when sequential burst.

even under fluctuating traffic conditions, helping to achieve optimal cross-DC transmission performance. Finally, the DQM algorithm inserts $\overline{R_{DQM}}$ in the ACK packet as the end-to-end recommended transmission rate.

$$dw = \begin{cases} dw + 1, & \text{if } token > 1 \\ dw - 1, & \text{if } token < 1 \end{cases} \tag{8}$$

$$\overline{R_{DQM}} = R_{credit} + \frac{dw \cdot MTU}{RTT_C} \tag{9}$$

3.3.2 *End-to-end Control Loop.*

The end-to-end control loop is used to feedback the results of the DQM algorithm to the sender. When the sender receives an ACK frame, the sender reads the $\overline{R_{DQM}}$ field. Combined with the near-source feedback loop, the sender calculates the final sending rate R_{MLCC} as:

$$R_{MLCC} = Min(R_{NS}, \overline{R_{DQM}}) \tag{10}$$

where R_{NS} is the rate calculated based on the near-source feedback signal.

3.4 Bringing Things Together

MLCC achieves rapid congestion relief by using near-source feedback and receiver-driven control loops to independently adjust traffic rates within datacenters in the short term. On top of that, MLCC also introduces an end-to-end feedback mechanism and DQM algorithm to coordinate rate control between datacenters, ensure long-term rate stability, and effectively manage the queues of several DCI-switches. With the above control loops and algorithms, MLCC is better able to ensure fairness in mixed traffic and control end-to-end latency.

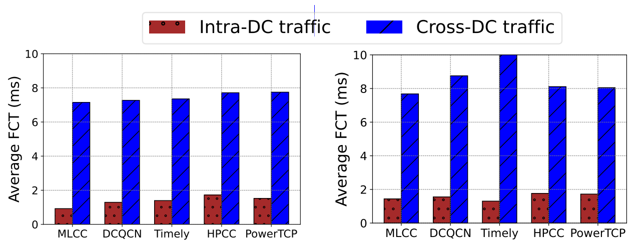
4 Evaluation

In this section, we use the NS-3 simulator [21] to evaluate the performance of MLCC and compare it with existing RDMA congestion control algorithms . We first test the convergence and buffer occupancy of MLCC. Then we conduct large-scale tests under different network settings.

4.1 Setup

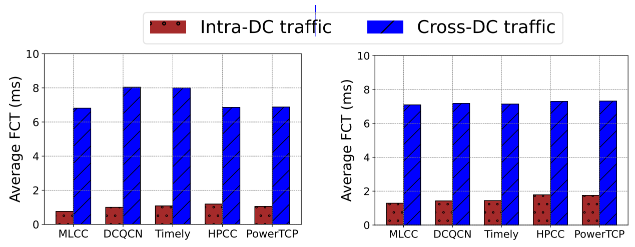
Simulation topology: The topology in the NS-3 simulation uses the same spine-leaf structure as shown in Fig. 1. Each datacenter has two spines and four leaves. The capacity of all switch-to-switch links is 100 Gbps, and the capacity of server-to-switch links is 25 Gbps, leading to 4 : 1 oversubscription similar to prior work [24]. The propagation delay of server-to-switch is 1 μs, and the propagation delay of other links in the datacenter is 5 μs. The propagation delay of long-distance links is 3 ms. The buffer sizes of the switches within the datacenter and the DCI-switches are 22 MB and 128 MB, respectively.

Testbed: We built a dumbbell testbed consisting of two DCI switches and two Tor switches. Each ToR switch is connected to 2 servers, each server equipped with a Mellanox 100Gbps NIC. The Tor switch uses a P4 switch with INT enabled [15]. Since we do not have MLCC based on RNIC hardware, we use eXpress Data Path (XDP) to implement MLCC features on the server and DCI switch [11, 27]. We evaluated the performance of DCQCN and MLCC on testbed.



(a) Avg. FCT of Websearch traffic (b) Avg. FCT of Hadoop traffic

Figure 11: High-load scenario: Intra-DC traffic at 50% load + Cross-DC traffic at 20% load.



(a) Avg. FCT of Websearch traffic (b) Avg. FCT of Hadoop traffic

Figure 12: Light-load scenario: Intra-DC traffic at 30% load + Cross-DC traffic at 10% load.

Workload: We generated traffic using Websearch [3] and Hadoop [23] traffic distributions to evaluate our algorithm. Intra-datacenter traffic loads and cross-datacenter traffic loads are set between 20% and 50%.

Parameter setting: We compare MLCC with DCQCN [32], Timely [20],HPCC [16] and PowerTCP [2]. For the DQM algorithm, we set θ and D_t to 18 ms and 1 ms respectively. We set m and α at 5 and 0.5, respectively. We set the parameters for DCQCN following the suggestion in [16]. For HPCC and PowerTCP, we use the native recommended parameters, respectively.

4.2 Convergence

Fig. 7 illustrates the convergence behavior under different startup methods when the congested link is located in the sender-side datacenter. Whether flows are initiated simultaneously or sequentially, MLCC quickly stabilizes to a fair bandwidth allocation. Fig. 8 depicts the convergence of various startup methods when the congested link is in the receiver-side datacenter. As shown in the figure, once the fair rate is reached, if the queue delay at the receiver-side DCI-switch exceeds the threshold, the DQM algorithm gradually reduces the sending rate, eventually reconverging to the fair rate. These experiments show that MLCC converges quickly and exhibits strong stability. In addition, MLCC reduces the queue length of the DCI-switch by adjusting the sender's rate after convergence.

4.3 DCI-switch Buffer Occupancy

When D_t is set to 1 ms, we evaluated the performance of the DQM algorithm under different θ values. The goal of the DQM algorithm is to smoothly set the queue length to the target value in θ time. Therefore, we choose to set the value of θ to the multiple of the RTT_C, 6 ms, 18 ms, 30 ms respectively. As shown in Fig. 9a, when multiple flows start simultaneously, DQM is able to quickly regulate the queue length, reducing it from a peak of 40 MB to around 5 MB. With $\theta = 6$ ms, the DQM algorithm behaves more aggressively, resulting in some jitter. In contrast, when $\theta = 30$ ms, the queue length converges slowly, while at $\theta = 18$ ms, convergence is relatively faster.

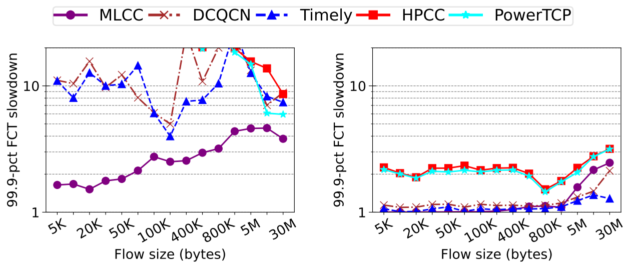
As shown in Fig. 9b, when D_t=1 ms and θ = 18 ms, DQM effectively controls the queue length of each flow within the targeted queuing delay. The queue length of each flow decreases from approximately 10 MB to about 1.5 MB, which aligns with the fair transmission rate of each flow—about 12.5 Gbps—since 12.5 Gbps $\times D_t = 1.5$ MB. In Fig. 10, when multiple flows are transmitted sequentially, DQM rapidly reduces the queue length from a peak of 25 MB to around 5 MB. After maintaining the queue for a while, as multiple cross-datacenter flows complete, the queue gradually empties.

4.4 Large-scale Simulations

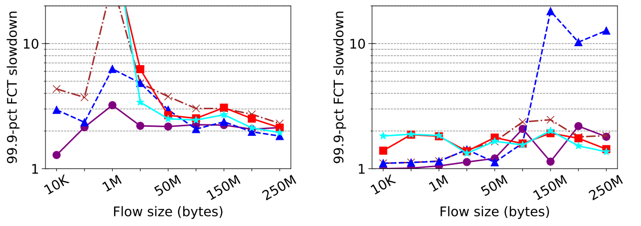
Heavy Load Scenario. Heavy load refers to 50% intra-datacenter traffic and 20% cross-datacenter traffic. Fig. 11 presents the average Flow Completion Time (FCT) for different traffic patterns under heavy load conditions. MLCC effectively reduces the FCT for both intra-datacenter and cross-datacenter flows across various traffic patterns. Specifically, under the Websearch traffic pattern, MLCC reduces the average FCT of intra-datacenter traffic by 28%, 33%, 46%, and 39% compared to DCQCN, Timely, HPCC, and PowerTCP, respectively. For the Hadoop traffic pattern, MLCC reduces the average FCT of intra-datacenter flows by 7%, 10%, 18%, and 16% compared to DCQCN, Timely, HPCC, and PowerTCP, respectively. Regarding cross-datacenter traffic, MLCC reduces the average FCT by 12%, 41% 5%, and 4%, respectively, compared with DCQCN, Timely, HPCC, and PowerTCP. We found that Timely is a delay-based algorithm, which easily causes data flow starvation, resulting in long FCT time. Although MLCC intervenes in cross-datacenter congestion control early by shortening the control loop, it ensures improvements in the average FCT for both intra- and cross-datacenter flows.

Fig. 13 depicts the 99.9th percentile FCT. As shown in Fig. 13a, MLCC reduces FCT for intra-datacenter flows across nearly all flow sizes under the Websearch traffic pattern. In Fig. 13b, MLCC reduces the FCT for cross-datacenter flows when the flow size is below 5 MB. However, for flows larger than 5 MB, MLCC slightly increases the FCT of cross-datacenter flows. This is because other algorithms rely on end-to-end congestion signal feedback and cannot effectively manage the bursts of cross-datacenter flows, while MLCC intervenes proactively to ensure fairness among mixed traffic. The results under the Hadoop traffic pattern are consistent with those observed for the Websearch traffic pattern.

Light Load Scenario. Light load refers to 30% intra-datacenter traffic and 10% cross-datacenter traffic. As shown in Fig. 12, MLCC consistently reduces the average FCT for both intra-datacenter and cross-datacenter flows across different traffic patterns. Specifically, under the Websearch traffic pattern, MLCC reduces the average FCT of intra-datacenter traffic by 24%, 30%, 36%, and 27%, compared to DCQCN, Timely, HPCC, and PowerTCP, respectively. Under the

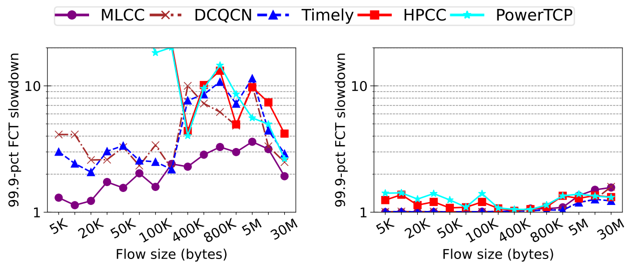


(a) Intra-DC traffic of Websearch (b) Cross-DC traffic of Websearch

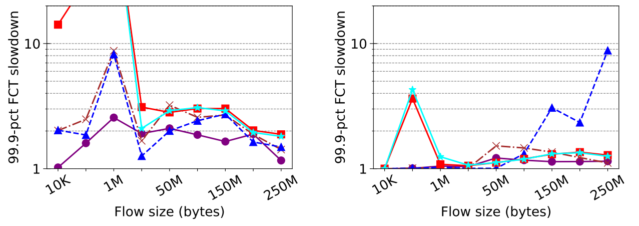


(c) Intra-DC traffic of Hadoop (d) Cross-DC traffic of Hadoop

Figure 13: 99.9% FCT of heavy-load scenario.



(a) Intra-DC traffic of Websearch (b) Cross-DC traffic of Websearch



(c) Intra-DC traffic of Hadoop (d) Cross-DC traffic of Hadoop

Figure 14: 99.9% FCT of light-load scenario

Hadoop traffic pattern, MLCC reduces the average FCT of intra-datacenter traffic by 9%, 10%, 27%, and 26%, respectively, compared to DCQCN, Timely, HPCC, and PowerTCP.

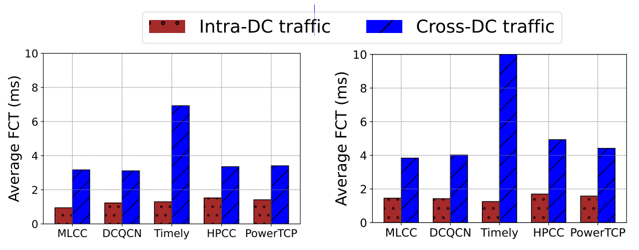
Fig. 14 presents the 99.9th percentile FCT for different algorithms under light load scenarios. Across both the Websearch and Hadoop traffic patterns, MLCC reduces the FCT for intra-datacenter flows across almost all flow sizes. As shown in Fig. 14b, MLCC reduces the FCT for cross-datacenter flows when the flow size is below 5 MB. However, for flow sizes larger than 5 MB, MLCC marginally increases the FCT for cross-datacenter flows. This behavior is consistent with the heavy load scenario, where MLCC's proactive intervention in congestion control ensures fairness among mixed flows but slightly increases completion times for larger flows.

4.5 Different topologies

We change the long-distance link latency to 1 ms. Fig. 15 shows the average FCT of different traffic patterns under heavy load. We can see that the average FCT of the existing scheme is quite large. In MLCC, the sender DCI-switch quickly returns the corresponding congestion information to the sender and reduces the queue latency of the receiver-side DCI-switch. Therefore, in the Websearch traffic pattern, MLCC reduces the average FCT of intra-DC flows by 22% compared to DCQCN. In the Hadoop traffic pattern, MLCC reduces the average FCT of cross-DC flows by 5% compared to DCQCN.

4.6 Testbed Experiments

We conducted tests using the Hadoop traffic model in a testbed environment. Fig. 16 presents the results. As we can see, MLCC is able to improve both intra-datacenter and inter-datacenter traffic. Specifically, compared to DCQCN, MLCC improves the overall average FCT by 19.3%. The experimental results show that MLCC can improve the performance of data flows in the cross-datacenter environment.



(a) Avg. FCT of Websearch traffic (b) Avg. FCT of Hadoop traffic

Figure 15: Avg. FCT of heavy-load scenario while cross-DC link delay is 1ms.

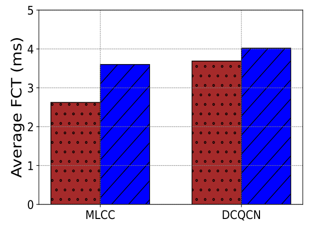


Figure 16: The Avg. FCT of testbed.

5 Discussion

Deployability of MLCC: Existing programmable switches not only support the INT feature, but also allow precise control of individual queue rates. TCP-INT software is a P4-based telemetry system that reduces signaling overhead by inserting telemetry data directly into the packet [11, 22]. Advanced Flow Control (AFC) on the P4 switches can dynamically adjust the queue transmission rate to alleviate network congestion. Therefore, in terms of deployability, existing technologies can fully support the deployment of MLCC.

In summary, our framework not only supports existing CCA, but also responds quickly and accurately to network changes.

6 Conclusion

Traditional datacenter congestion control protocols cannot cope with the challenge of cross-datacenter transmission. This paper proposes MLCC, a cross-datacenter transmission protocol based on minimization control loops to address this challenge. The MLCC consists of two main control loops: the short-term intra-data center control loop and the long-term inter-datacenter control loop. The short-term loop uses near-source feedback and receiver-side driving algorithms to relieve congestion in the datacenters. The long-term end-to-end control loop integrates feedback from the short-term loop and DCI-switch queue management algorithm to adjust send rates to stabilize traffic and reduce DCI-switch queue lengths. Together, these loops enable efficient, localized, and global traffic management, ensuring rapid congestion relief and long-term stability. Moreover, MLCC can be compatible with existing methods on different loops. Large-scale simulations show that MLCC effectively improves the performance of intra-datacenter and cross-datacenter traffic.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable feedback on this paper. This work was partially supported by National Key R&D Program of China No.2024YFB2908700, Pengcheng Laboratory Cooperation Project No.2025KF1A0010 and BMFTR in Germany Joint 6G-RIC Project No.16KISK0-PIN.

References

[1] 2023. Empowering Azure Storage with RDMA. In *NSDI 23*. USENIX Association, Boston, MA, 49–67.

[2] Vamsi Addanki, Oliver Michel, and Stefan Schmid. 2022. {PowerTCP}: Pushing the performance limits of datacenter networks. In *NSDI 22*. 51–70.

[3] Mohammad Alizadeh, Albert Greenberg, David A Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. 2010. Data center tcp (dctcp). In *Proceedings of the ACM SIGCOMM 2010 Conference*. 63–74.

[4] Ran Ben Basat, Sivaramakrishnan Ramanathan, Yuliang Li, Gianni Antichi, Minian Yu, and Michael Mitzenmacher. 2020. PINT: Probabilistic in-band network telemetry *(SIGCOMM '20)*. 662–680.

[5] Yanqing Chen, Chen Tian, Jiaqing Dong, Song Feng, Xu Zhang, Chang Liu, Peiwen Yu, Nai Xia, Wanchun Dou, and Guihai Chen. 2022. Swing: Providing long-range lossless rdma via pfc-relay. *IEEE Transactions on Parallel and Distributed Systems* 34, 1 (2022), 63–75.

[6] Vojislav Dukic, Ginni Khanna, Christos Gkantsidis, Thomas Karagiannis, Francesca Parmigiani, Ankit Singla, Mark Filer, Jeffrey L. Cox, Anna Ptasznik, Nick Harland, Winston Saunders, and Christian Belady. 2020. Beyond the mega-data center: networking multi-data center regions *(SIGCOMM '20)*. Association for Computing Machinery, New York, NY, USA, 765–781.

[7] Adithya Gangidi, Rui Miao, Shengbao Zheng, Sai Jayesh Bondu, Guilherme Goes, Hany Morsy, Rohit Puri, Mohammad Riftadi, Ashmitha Jeevaraj Shetty, Jingyi Yang, et al. 2024. RDMA over Ethernet for Distributed Training at Meta Scale. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 57–70.

[8] Yixiao Gao, Qiang Li, Lingbo Tang, Yongqing Xi, Pengcheng Zhang, Wenwen Peng, Bo Li, Yaohui Wu, Shaozong Liu, Lei Yan, et al. 2021. When cloud storage meets {RDMA}. In *NSDI 21*. 519–533.

[9] Yantao Geng, Han Zhang, Xingang Shi, Jilong Wang, Xia Yin, Dongbiao He, and Yahui Li. 2023. Delay Based Congestion Control for Cross-Datacenter Networks. In *2023 IWQoS*. IEEE, 1–4.

[10] Mark Handley, Costin Raiciu, Alexandru Agache, Andrei Voinescu, Andrew W. Moore, Gianni Antichi, and Marcin Wójcik. 2017. Re-architecting datacenter networks and stacks for low latency and high performance *(SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 29–42.

[11] Jörn-Thorben Hinz, Vamsi Addanki, Csaba Györgyi, Theo Jepsen, and Stefan Schmid. 2023. TCP's Third Eye: Leveraging eBPF for Telemetry-Powered Congestion Control. In *Proceedings of the 1st Workshop on eBPF and Kernel Extensions*. 1–7.

[12] Ninad Hogade and Sudeep Pasricha. 2022. A survey on machine learning for geo-distributed cloud data center management. *IEEE Transactions on Sustainable Computing* (2022).

[13] Kevin Hsieh, Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R Ganger, Phillip B Gibbons, and Onur Mutlu. 2017. Gaia:{Geo-Distributed} machine learning approaching {LAN} speeds. In *NSDI 17*. 629–647.

[14] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Venkata, et al. 2013. B4: Experience with a globally-deployed software defined WAN. *ACM SIGCOMM Computer Communication Review* (2013).

[15] Grzegorz Jereczek, Theo Jepsen, Simon Wass, Bimmy Pujari, Jerry Zhen, and Jeongkeun Lee. 2022. Tcp-int: lightweight network telemetry with tcp transport. In *Proceedings of the SIGCOMM'22 Poster and Demo Sessions*. 58–60.

[16] Yuliang Li, Rui Miao, Hongqiang Harry Liu, Yan Zhuang, Fei Feng, Lingbo Tang, Zheng Cao, Ming Zhang, Frank Kelly, Mohammad Alizadeh, et al. 2019. HPCC: High precision congestion control. In *Proceedings of the ACM special interest group on data communication*. 44–58.

[17] Hwijoon Lim, Jaehong Kim, Inho Cho, Keon Jang, Wei Bai, and Dongsu Han. 2023. FlexPass: A Case for Flexible Credit-based Transport for Datacenter Networks. In *Proceedings of the Eighteenth European Conference on Computer Systems*. 606–622.

[18] Hwijoon Lim, Juncheol Ye, Sangeetha Abdu Jyothi, and Dongsu Han. 2024. Accelerating Model Training in Multi-cluster Environments with Consumer-grade GPUs. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 707–720.

[19] Changyun Luo, Huaxi Gu, Lijing Zhu, and Huixia Zhang. 2024. FlowStar: Fast Convergence Per-Flow State Accurate Congestion Control for InfiniBand. *IEEE/ACM Transactions on Networking* (2024).

[20] Radhika Mittal, Vinh The Lam, Nandita Dukkipati, Emily Blem, Hassan Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, and David Zats. 2015. TIMELY: RTT-based congestion control for the datacenter. *ACM SIGCOMM Computer Communication Review* 45, 4 (2015), 537–550.

[21] NS-3. [n. d.]. NS-3 Network Simulator, Version 3.36. https://www.nsnam.org/releases/ns-3-36/. 2024.

[22] Konstantinos Papadopoulos, Panagiotis Papadimitriou, and Chrysa Papagianni. 2021. Pfa-int: Lightweight in-band network telemetry with per-flow aggregation. In *2021 NFV-SDN*. IEEE, 60–66.

[23] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C Snoeren. 2015. Inside the social network's (datacenter) network. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. 123–137.

[24] Ahmed Saeed, Varun Gupta, Prateesh Goyal, Milad Sharif, Rong Pan, Mostafa Ammar, Ellen Zegura, Keon Jang, Mohammad Alizadeh, Abdul Kabbani, and Amin Vahdat. 2020. Annulus: A Dual Congestion Control Loop for Datacenter and WAN Traffic Aggregates *(SIGCOMM '20)*. New York, NY, USA, 735–749.

[25] Danfeng Shan, Yuqi Liu, Tong Zhang, Yifan Liu, Yazhe Tang, Hao Li, and Peng Zhang. 2023. Less is more: Dynamic and shared headroom allocation in pfc-enabled datacenter networks. In *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 591–602.

[26] Daniel Tauber, Brian Smith, David Lewis, Ernest Muhigana, Morten Nissov, Donald Govan, JinLin Hu, Yuxin Zhou, Jian Wang, Wen-Jr Jiang, et al. 2023. Role of coherent systems in the next DCI generation. *Journal of Lightwave Technology* 41, 4 (2023), 1139–1151.

[27] Marcos A. M. Vieira, Matheus S. Castanho, Racyus D. G. Pacífico, Elerson R. S. Santos, Eduardo P. M. Câmara Júnior, and Luiz F. M. Vieira. 2020. Fast Packet Processing with eBPF and XDP: Concepts, Code, Challenges, and Applications. *ACM Comput. Surv.* 53, 1, Article 16 (Feb. 2020), 36 pages.

[28] Zirui Wan, Jiao Zhang, Mingxuan Yu, Junwei Liu, Jun Yao, Xinghua Zhao, and Tao Huang. 2024. Bicc: Bilateral congestion control in cross-datacenter rdma networks. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*. IEEE, 1381–1390.

[29] Siyu Yan, Xiaoliang Wang, Xiaolong Zheng, Yinben Xia, Derui Liu, and Weishan Deng. 2021. ACC: Automatic ECN tuning for high-speed datacenter networks. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. 384–397.

[30] Peiwen Yu, Feiyang Xue, Chen Tian, Xiaoliang Wang, Yanqing Chen, Tao Wu, Lei Han, Zifa Han, Bingquan Wang, Xiangyu Gong, et al. 2023. Bifrost: Extending RoCE for Long Distance Inter-DC Links. In *2023 ICNP*. IEEE, 1–12.

[31] Renjie Zhou, Dezun Dong, Shan Huang, and Yang Bai. 2021. FastTune: Timely and Precise Congestion Control in Data Center Network. 238–245.

[32] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. 2015. Congestion control for large-scale RDMA deployments. *ACM SIGCOMM Computer Communication Review* 45, 4 (2015), 523–536.

[33] Shaojun Zou, Jiawei Huang, Jingling Liu, Tao Zhang, Ning Jiang, and Jianxin Wang. 2021. Gtcp: Hybrid congestion control for cross-datacenter networks. In *2021 ICDCS*. IEEE, 932–942.